An Information System for Component Building

by Theodore H. Myer

Bolt, Beranek & Newman, Inc.

Abstract

An information system for building design is explored as a case history, with emphasis on the characteristics most important to the user. Experience with the system suggested that key criteria are the effectiveness with which designs are represented in the computer, and the ease with which users can work with the stored material. The design of a simple information system that appears to meet these criteria is described.

I. Introduction

This paper describes research carried out in collaboration with Carl Koch and Associates, Architects, under the sponsorship of the Urban Development Corporation of New York State. The project grew from a desire to apply computer technology to the information problems of component building design.

Our working context was Carl Koch's Techcrete building system. Techcrete is based on precast floor planks and bearing wall panels; post tensioning techniques enable construction of high- as well as low-rise buildings. Our partcular focus in this project was the use of Techcrete in the design of a mixed public and private housing complex of about 1000 units for a site near New York City (1).

Within this context we wanted to investigate a computer system capable of managing design information and supplying a number of separate retrieval, analytic, and reporting functions. We approached this by developing a limited initial system, putting it into practical use, and letting our experience in using the system guide its further development. By this means we gained not only a practical working tool, but also some insight into the nature and applicability of information systems in architecture.

In the work, our greatest concern was to find an effective way to represent building designs in the computer. The representation would have to carry information satisfying the various functions to be included in the system. More importantly, we felt, it would have to be convenient to work with. This suggested ease of assembly and editing as key criteria, and most importantly "naturalness" in the sense of reflecting the designer's own ways of thinking about and organizing architectural material.

What resulted was an initial system with incremental cost estimation as its main function. Once in use, the system became more general. At this writing it includes or will shortly include reporting functions for net and gross areas, heat and cooling loads, and various schedules. Also, during use the system's data base evolved into a cataloguing and control tool for Techcrete, and a pool of design information upon which to draw in establishing new designs. At present we are adding a graphic capability and plan to extend the system to aid in manufacturing and construction.

II. The Initial System

We selected construction cost estimation as a first goal, since we felt it would require an initial system with most of the important components needed in a subsequent, more elaborate system. Within the general framework

of cost estimation, we were particularly concerned with incremental estimation (2,3). That is, we wanted the designer to be able to test design changes for their effect on construction cost, in a continuous manner throughout the design process. That goal was met by building an initial system with four main components:

1) A representation for building designs.
2) A library of building components containing cost and other information about each component. The component library provides a base for building the design representation, and a source of cost data for estimation.
3) A series of commands for assembling and modifying the design representation.
4) Computer programs that perform quantity surveys, compute costs, and issue cost reports.

Figure 1 shows the functional relationship of these four components. Physically, the system resides in a DEC PDP-10 time-shared computer, under control of the TENEX time-sharing system under development at BBN. Communication and data input take place through remote Teletype terminals. Reports can be output via Teletype or line printer.

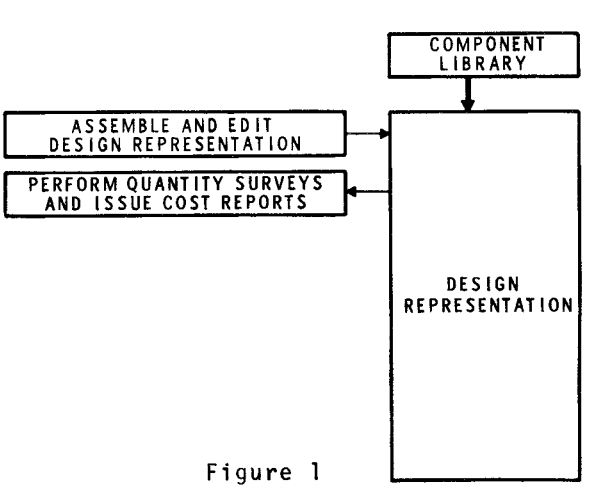

Figure 1

To represent buildings in the computer, we selected a tree structure (4). Similar techniques for organizing design information have been used by a number of other workers (5,6,7,8). Carr (5), in particular, gives a lucid explanation. We chose a tree structure as likely to meet our demands for ease of manipulation and "naturalness". In addition, the tree quite easily supports cost computation.

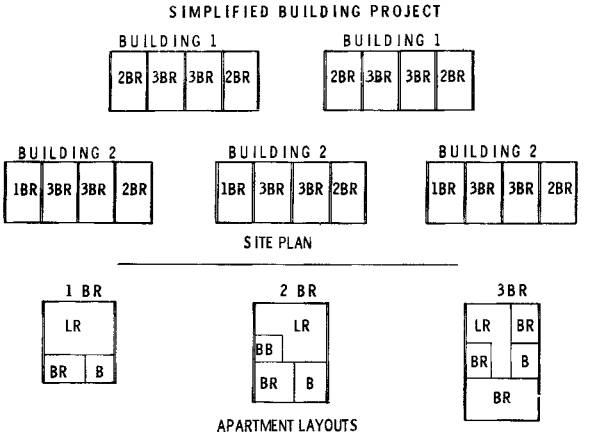

Figure 2

To illustrate how a building project can be mapped into a tree representation in the computer, Figure 2 shows a grossly simplified building Project and Figure 3 shows its representation in computer memory. As indicated in Figure 3, the tree is made up of nodes and branches, with nodes standing for groups of things and branches pointing to other nodes to indicate what and how much of it belongs in each group. For example, the top node, named "PROJECT-1", stands for the entire project and is made up of five buildings - two of type 1 and three of type 2. Each building in turn is a group of apartments. Each apartment comprises lower-level groupings accounting for its interior components and structural envelope.

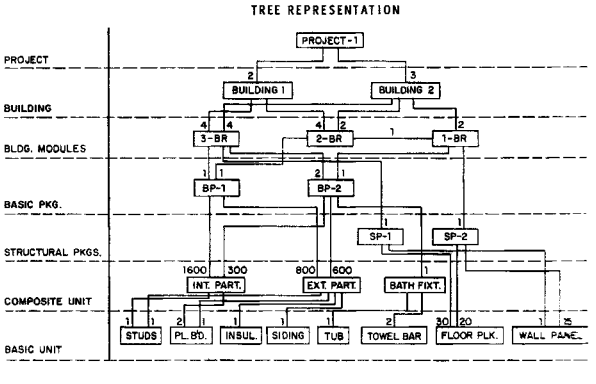

Figure 3

This aggregation into groups continues down through the structure, until at the bottom one reaches the elementary components from which the project is built.

In computer memory, nodes comprise data blocks. Within a node, branches are represented by pointers to other nodes. Each branch has an associated number indicating how much of the second node is contained in the first. The tree structure permits rather efficient use of computer memory because, as with multi-level "instances" in computer graphics, or nested subroutine calls in ordinary programming, each component or assembly is defined just once and then referred to or "called" when needed.

As well as being compact, the tree structure allows considerable flexibility. Though Figure 3 suggests seven levels and specific uses, the designer can employ as many levels as wanted for whatever purposes he has in mind.

Although most branches in Figure 1 descend one level, there is nothing to prevent branches that descend multiple levels or none at all. This permits low-level components or groups to be "tacked on" to groups much higher in the structure. For example, exterior light fixtures (at the bottom level) might be added to the project (at the top) in this way. This also makes it easy to create slightly modified versions of standard groupings. For example, a modified three-bedroom apartment might reference the standard version and be augmented by a few extra square feet of interior partition.

Closely related to this augmentation capability is the possibility of subtracting from a standard by using negative branch quantities. This proved useful in the actual project in permitting modified versions containing fewer of one component or another than the original.

Finally, although the tree in Figure 3 ascends to a single node at the top, there is nothing against multiple top nodes, and this is sometimes useful. With the actual project, this permitted useful subgroupings of the buildings. For example, one collection of "top" nodes called out groups of buildings segregated by height, while another pair of nodes separated the project into public and private sectors. These partitionings of the data were useful in serving design and legal needs of cost estimation.

Thus the central component in the information system is a flexible and rather loosely organized tree structure, to be approached very directly by the designer in formulating design representations.

As suggested by Figure 3, elements of the system's second major ingredient, the building component library, form the bottom of the tree. Actually, this data is kept in a separate part of computer memory and referred to by branch pointers in the nodes that lie above it. In the initial system the library contained an identifying code for each component, a text description, the CSI and FHA categories to which the component belongs, and certain temporary storage registers to hold component quantities during estimation.

The component library forms the most permanent part of the data base, changing only as the building system itself is changed. In contrast, the tree structure that represents a given building project will grow and change as the design is built up and successively modified. Furthermore, at any given time, the component library may support any number of separate tree structures, each describing a different project.

INITIAL SYSTEM COMMANDS

DATA ENTRY AND EDITING

DEFINE NODE___LEVEL___
BRANCH QUANTITY

KILL NODE____
PUT BRANCH___ON NODE___QUANTITY___
TAKE BRANCH___FROM NODE___
REPLACE BRANCH___OF NODE___WITH BRANCH____
QUANTIFY BRANCH___OF NODE___QUANTITY____

ESTIMATION AND LISTING

ESTIMATE STARTING AT___REG/INV___FORMAT___SORT____
REPORT COSTS FORMAT___SORT___
LIST FROM LEVEL___TO LEVEL___
LIST NODE____

Figure 4

To assemble and manipulate design representations, the system includes simple commands (Figure 4) that refer directly to tree elements. These commands govern the creation and destruction of nodes and the

modification of node contents. To create a node (Define) one specifies its level, names it, and then lists the branches and their quantities that form its contents. To remove a node (Kill) one need only give its name. To change node contents, one can add (Put) or remove (Take) branches. Although these four commands suffice for all functions, two more were included for convenience. Replace is equivalent to Take followed by Put; Quantity allows one to change quantity without otherwise affecting node contents. As an added convenience, Take, Replace, and Quantity can modify all nodes, if desired, rather than a single named node. This allows one to do such things as replace all occurrences of one building component with another.

Internally, the assembly and editing commands are backed up by appropriate indexing, retrieval, and storage management programs. As editing occurs these programs access and modify the tree structure.

The remaining commands in Figure 4 govern the estimating and reporting functions that form the fourth component of the initial system. To perform conventional cost estimates, a simple "tree-walking" program performs the equivalent of a quantity survey on the tree structure. It does this by tracing down all possible branches, collecting branch quantities as it goes, and accumulating them in temporary registers at the bottom of the tree. A reporting program then computes and tabulates component and total costs.

The tree-walking program can be started off at any point in the tree. If started at the top, it will yield an estimate for the whole project. If started somewhere else, it will yield a quantity survey and cost estimate for the node selected. Figure 5 shows two reports for the tree shown in Figure 3, one covering the whole project and one covering just the one bedroom apartment. These were produced by starting the program at nodes Project-1 and 1-BR respectively. This ability to begin the takeoff process anywhere permitted separate estimates of individual apartments, buildings, and other design subgroupings of the project.

Figure 5 shows just one of several possible report formats. Component tabulations can be sorted, if desired, by CSI or FHA categories. As well as detailed tabulations, summary reports can be produced giving category totals or just the final total.

COST REPORT -- UNIT PRJ1 PAGE 1

NAME	DESCRIPTION	QUANTITY	MATERIAL COST UNIT	MATERIAL COST TOTAL	LABOR COST UNIT	LABOR COST TOTAL
STUD	2+1/2" MET STD (S. F)	171600	.32	54912	0.00	0
PLBD	1/2" GYP BD	280000	.08	22400	.13	36400
INSL	2" BATT INSULATION	63200	.05	3160	.05	3160
SIDG	METAL SIDING	63200	.27	17064	.20	12640
TUB	BATHTUB	60	35.00	2100	20.00	1200
TBAR	TOWEL BAR	120	1.00	120	.50	60
WALL	PRECAST WALL PANEL	50	864.00	43200	432.00	21600
FPLK	PRESTRESSED FLR PLK	1000	90.00	90000	47.00	47000
TOTAL BASIC COSTS				232956		122860
INSURANCE & TAXES ON LABOR		@ 30%		36618		
TOTAL LABOR				122860		
SUBTOTAL				391634		
GENERAL CONDITIONS		@ 5%		19582		
SUBTOTAL				411216		
OVERHEAD & PROFIT		@ 10%		41122		
SUBTOTAL				452337		
BOND		@ 1%		4523		
GRAND TOTAL COST UNIT PRJ1				456861		

COST REPORT -- UNIT 1-BR PAGE 1

NAME	DESCRIPTION	QUANTITY	MATERIAL COST UNIT	MATERIAL COST TOTAL	LABOR COST UNIT	LABOR COST TOTAL
STUD	2+1/2" MET STD (S. F)	1500	.32	480	0.00	0
PLBD	1/2" GYP BD	2400	.08	192	.13	312
INSL	2" BATT INSULATION	600	.05	30	.05	30
SIDG	METAL SIDING	600	.27	162	.20	120
TUB	BATHTUB	1	35.00	35	20.00	20
TBAR	TOWEL BAR	2	1.00	2	.50	1
WALL	PRECAST WALL PANEL	2	864.00	1296	432.00	648
FPLK	PRESTRESSED FLR PLK	20	90.00	1800	47.00	940
TOTAL BASIC COSTS				3997		2071
INSURANCE & TAXES ON LABOR		@ 30%		621		
TOTAL LABOR				2071		
SUBTOTAL				6689		
GENERAL CONDITIONS		@ 5%		334		
SUBTOTAL				7024		
OVERHEAD & PROFIT		@ 10%		702		
SUBTOTAL				7726		
BOND		@ 1%		77		
GRAND TOTAL COST UNIT 1-BR				7803		

Figure 5

The downward direction of the quantity survey program suggested reversing the procedure, starting with unit costs at the bottom, and extending costs and quantities up through the structure above. In contrast to the ordinary estimate, which analyzes a single node in detail, this procedure computes the total cost of each node in the tree. We called the resulting output an "inverted" cost report because the processing involved is, loosely speaking, the inverse of the ordinary case. In practice, inverted reports provide a convenient tool for analyzing the relative cost contributions of different parts of a project. For example, one such report shows, for every node, the cost contributed by each of its branches.

The Estimate command starts either of the two estimating processes and allows one to select an initial report format. Report Costs permits additional reports on the data generated in a prior estimate. The

List commands provide for tabulations of tree and component library contents.

III. Practical Experience and Extensions

Once complete, we put the system to work, and tested it during several stages in the design development of the housing project. Figure 6 shows a model photograph of the project; its 1000-odd units were grouped into three-, six-, ten-, and eighteen-story buildings. Inside the computer, the project required a tree of 10 levels and about 200 nodes.

As expected, the system's chief value was its ability to do incremental and comparative estimation. It was possible to test such things as differing apartment layouts and alternative groupings of apartments into buildings, for their effect on total project cost. It was also useful to compute unit apartment and building costs, and to compare costs for public and private sectors. After each series of definitive design changes, we ran a complete series of cost reports, which formed one basis for subsequent design work.

Figure 6

During this testing phase we began to add new functions. We have completed report functions that tabulate floor areas and compute total areas for each type of precast panel used in a design. These functions use tree-processing techniques similar to the quantity survey. For example, to calculate floor areas, one traces the tree downward to the apartment level, and picks up unit areas stored on these nodes. We are currently working on heat and cooling load and schedule reports. To generate finish or door schedules, we will move across the tree at the apartment level, tracing down at each node to locate the desired items.

During testing, we watched the reactions of users with some concern. The tree representation exposes the user rather directly to the data structure inside the computer. It requires that the map design information from sketches and drawings into a hierarchical representation quite different in appearance. Further, he must keep the tree constantly in mind as the design progresses, editing it each time drawings are changed. We were concerned that all this would be burdensome enough to make the system unattractive.

In practice, this fear proved unfounded. With some practice, the tree representation and editing commands proved quite easy to use. In fact, some of the more powerful techniques for using the tree structure developed only after some experience with the system.

More importantly, the tree structure had a positive effect beyond its use in the computer. It developed into a tool that helped the designer organize and control design information outside the machine. Figures 7 and 8 make this somewhat clearer. In Figure 7, graphic information has been organized in a hierarchical fashion exactly paralleling the tree structure in the computer. Individual components on the left are carried through successive levels of aggregation leading to buildings on the right. Output, taken directly from the computer, tabulates the contents of each grouping; a uniform nomenclature has been used on the drawings and in the computer. Figure 8 carries this one step further, to the level of the site plan for the whole project.

What these montages show is the beginnings of a cataloguing system that keeps track of building components and their various levels of assembly into complete buildings. In part, this catalog resides in standard details and assembly drawings. Information inside the computer augments the drawings, indexing and organizing them, and quantifying their contents.

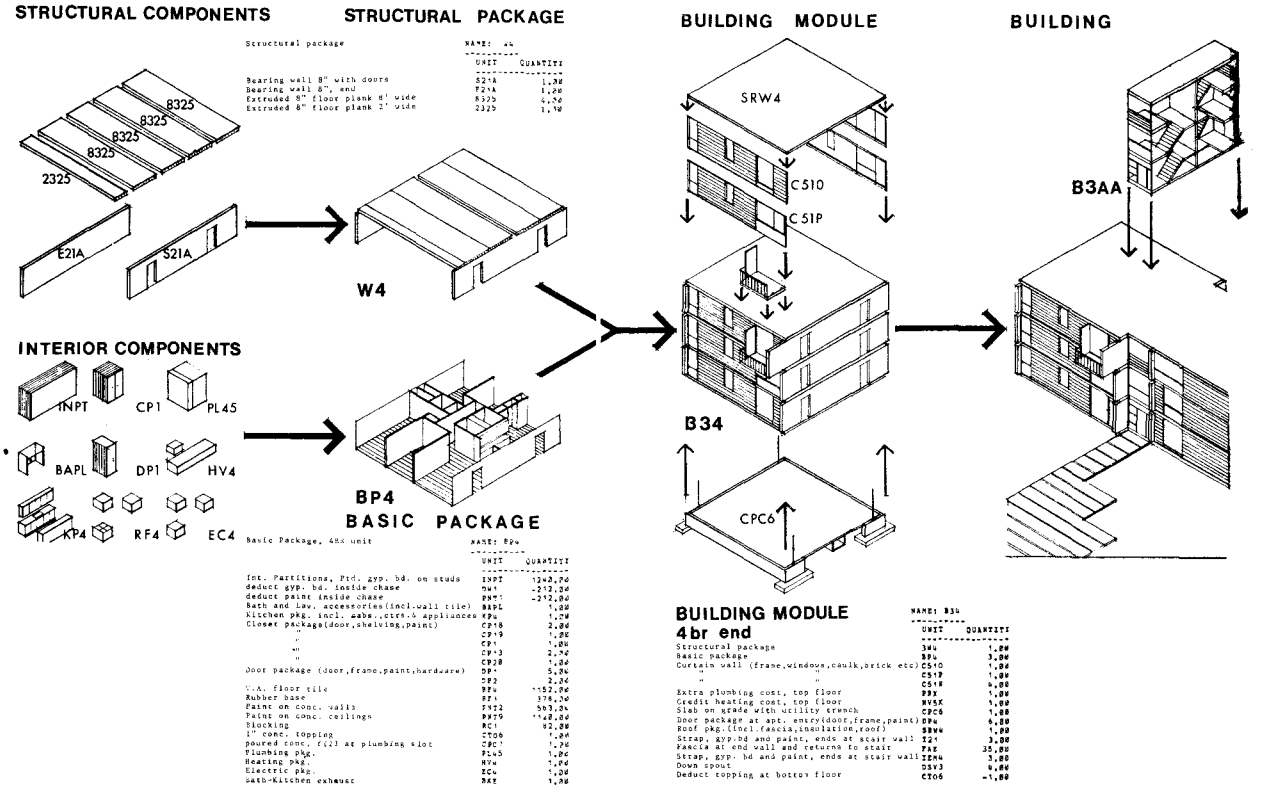

Figure 7

Although this "design catalog" was useful in the present project, it will be most valuable as a permanent control tool for Techcrete. Within the computer, this ongoing usage of design data has the effect of introducing a feedback loop, with completed design representations cycling back to become part of a pool of information upon which to draw in formulating new ones. This means several things. First, intuitively, it would seem to reflect rather naturally the evolving use of the building system in successive projects. The outcome of one project will influence the initial design of the next, and the feedback of design information in the computer reflects this quite directly.

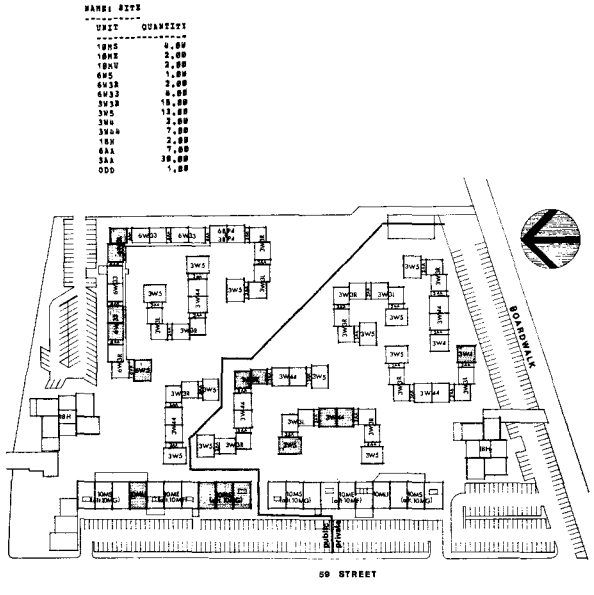

Figure 8

Second, feedback will speed up the process of assembling tree representations. In this initial project, it was necessary to start from scratch, defining every node, and building up the tree level by level until it was complete. In the future, we will be able to call in more or less complete "chunks" of tree structure to formulate an initial representation.

Third, the ability to start off with a fully developed tree will give us a uniformly detailed basis for cost estimation throughout the life of a

design project. This should overcome the discontinuity that occurs when a preliminary estimating basis (such as square footage) is dropped in favor of detailed quantity surveys (2). Figure 9 shows the system as extended to include additional report functions and the feedback of design information.

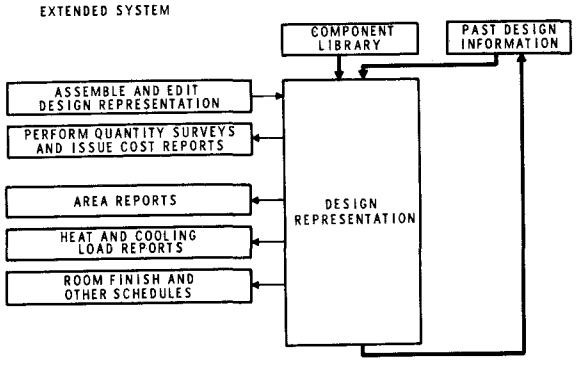

Figure 9

Inevitably, computer indexing of conventional drawings also suggested that we add a graphic capability to the system. In the beginning we had rejected this idea as jeopardizing our chances for practical results. However, by this time, we were pleased with how far we had got without graphics, and it seemed desirable to try some experiments.

Figures 10 and 11 show some preliminary results of this work. The material in these figures was assembled using an interactive display terminal,

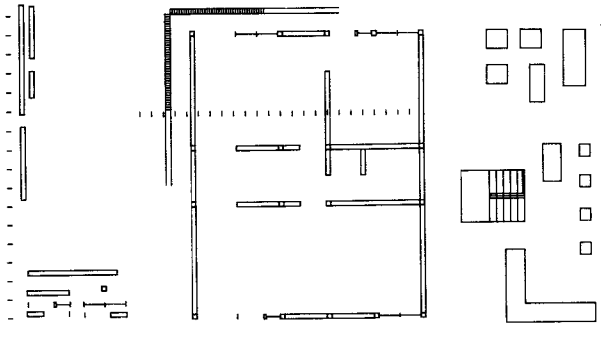

Figure 10

and then reproduced on a plotter. At the left in figure 11 we have created graphic symbols representing structural building components. In the center these have been grouped into the shell of a single apartment. At the right is a second group of components intended to form the apartment's interior contents.

Figure 11 shows a plan, at ground level, of a complete building. To get from figure 10 to figure 11, we combined the interior components with the structural shell to form a complete apartment. Then three of these were joined to form the building. The scales and tick marks in both figures are the computer equivalent of drafting tools; they help position picture elements as they are grouped together.

Internally, we used a tree structure similar to that in the initial system to store these drawings.

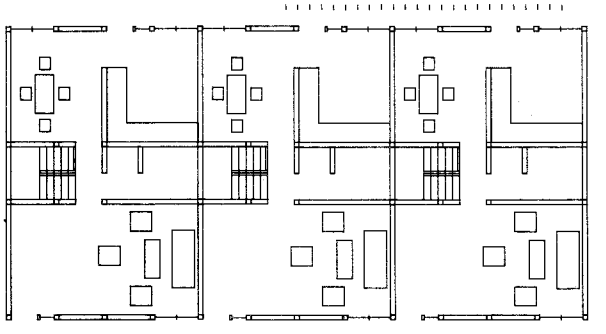

Figure 11

Individual components form the bottom level of the tree; groupings of these form nodes at successive levels above. Thus, figure 11 represents a single node at the top of the tree, containing three instances of the apartment node one level below plus an exterior wall. The only differences between this tree and the non-graphic one are 1) that component descriptions at the bottom level must contain graphic information describing their pictorial representations, and 2) that branches in the structure above must contain relative coordinates for the nodes they call out.

Externally, we exposed the user to the tree organization in much the same way as was done in the initial system. To produce figures 10 and 11, graphic symbols were first defined using a

stylus device in conjunction with the display. Next, instances of these were called up onto the screen and grouped together into successive levels of hierarchy until the pictures were complete.

In doing these experiments, we were not seeking to perform basic graphics research. Thus, for example, we used graphics support programs already available at BBN. Our goal, rather, was to explore the possibility of incorporating graphics into the system we had built, using already established techniques. Although the work is still experimental, there appear to be few difficulties in the way of putting it into practice.

IV. Future Extensions

So far we have considered system features carried at least to the point of experimental testing. What follows are some thoughts on extensions to be added in future work.

For one thing, there are several useful functions that would fit quite easily within the existing system framework. These include a number of engineering analysis and reporting tools. Without stretching the system too much one could at least partially automate the specification process by linking text passages to the entries in the component library. Somewhat more ambitiously, with added information and appropriate programs, the component library could be expanded into a retrieval and analysis subsystem that would assist in component evaluation and support further engineering functions.

Viewing the full scope of component building - from manufacturing through construction - one can foresee further extensions. If extended into manufacturing the system could take on many functions of information systems currently in use in industry - inventory control and production scheduling, to name just two. In construction, it could assist in progress and cost monitoring, and job scheduling. As well as yielding a broader system, the manufacturing, design, and construction functions would mutually reinforce each other. For example, inventory control could utilize pooled quantity surveys of all current projects. Conversely, direct access to manufacturing data would yield better unit costs and more accurate estimates.

Although developed for component building, we feel the system would also be effective with conventional structures such as housing, schools, and hospitals that are inherently hierarchical or repetitive. It probably wouldn't work too well, however, with amorphous or highly singular buildings such as opera houses or churches.

Although many functions could be supported by the present system, some jobs would require changes or extensions to its basic structure. Design generation, traffic simulation, and anything but the most rudimentary code checking all require topological information that can't be represented very well in a simple tree. Eastman (9) discusses other information structuring techniques that can be used in architectural problems.

V. Conclusion

This paper has explored the development of an information system for building design, focussing on its appearance to the user, and on the experience gained in using it. In this context, we feel the project demonstrates principally 1) the effectiveness with which simple tree structures can represent building projects, 2) the ease with which users can work directly with such structures, and 3) the extent to which a rather simple information system can assist in the design process.

Acknowledgement

The author wishes to thank Margaret Ross and Leon Lipshutz of Carl Koch and Associates, and Daniel Bobrow and William R. Sutherland of BBN for their contributions in support and guidance of this effort.

- Notes -

1. Techcrete is a registered trademark of Techrete Inc. Further information about the system can be obtained from Carl Koch and Associates, Inc., 35 Lewis Wharf, Boston 02110.

2. Myer, T. H., and R. I. Krauss, "Computer-Aided Cost Estimating

Techniques for Architects", Institute for Applied Technology, National Bureau of Standards, December 1965.

3. Krauss, R. I., and T. H. Myer, "Computer-Aided Cost Estimating Techniques" in Computer Applications in Architecture and Engineering, N. Harper Ed., New York: McGraw-Hill 1968.

4. Strictly speaking, we used a "semi-tree", since downward pointing branches can meet at nodes lower in the structure.

5. Carr, C. S., "Geometric Modeling", Computer Science Department, University of Utah, 1969.

6. Newman, William M., "An Experimental Program for Architectural Design", The Computer Journal, Vol. 9 No. 1, May 1966.

7. Cogswell, A. R., Werner Hausler, and C. David Sides, "Integrated Building Industry System", North Carolina Fund, 1968.

8. Davis, Charles F., "An Architectural Data Management System", to be published at this conference.

9. Eastman, Charles M., "Representations for Space Planning", Communications of the ACM, Vol. 13 No. 4, April 1970.